VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Aplikace metod kvadratického programování na ořezání zvukového záznamu v reálném čase

Quadratic programming methods applied on the real-time clipping of audiosignals

2014 Martin Moudrý

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání bakalářské práce

Student: **Martin Moudrý**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 1103R031 Výpočetní matematika

Téma: Aplikace metod kvadratického programování na ořezání zvukového záznamu v reálném čase
Quadratic programming methods applied on the real-time clipping of audiosignals

Zásady pro vypracování:

Ořezání (clipping) signálu v reálném čase je významnou operací v řadě audio aplikací. Tato operace ovšem redukuje kvalitu přehrávaného zvukového záznamu. Nedávno navržené ořezávací algoritmy založené na zachování kvality vnímání, významně převyšují standardní ořezávací techniky, co se týče výsledné kvality signálu. Tyto nové algoritmy vedou na řešení optimalizačních úloh s nerovnostními omezeními v reálném čase, což není vůbec snadné.

Cílem práce je nastudovat a integrovat do těchto nových algoritmů metody kvadratického programování dlouhodobě vyvíjené na katedře aplikované matematiky.

Seznam doporučené odborné literatury:

Podle doporučení školitele.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **prof. Ing. Tomáš Kozubek, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA OSTRAVA * Fakulta elektrotechniky a informatiky *

doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014

Moudrý
.....

Na tomto místě bych rád poděkoval prof. Ing. T. Kozubkovi, Ph.D. za jeho cenné rady a připomínky při vedení mé bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá řešením úlohy kvadratického programování. Popisuje algoritmy používané pro řešení této úlohy jak bez omezení, tak i s jednoduchými lineárními omezeními a aplikuje je na vybrané úlohy modelování průhybu struny a ořezání zvukového záznamu.

Klíčová slova: Optimalizační úloha, Metoda největšího spádu, Metoda sdružených gradientů, Barzilai-Borweinův algoritmus, MPRGP algoritmus, Modelování průhybu struny, Ořezání audiosignálu

Abstract

This bachelor thesis deals with quadratic programming problems. It describes algorithms, which are used for solving such problems without constraints and with simple linear inequality constraints. Thess algorithms are then used for modeling of string deflection and clipping of audiosignals.

Keywords: Optimization problem, Gradient descent, Conjugate gradient method, Barzilai-Borwein algorithm, MPRGP algorithm, Modeling of string deflection, Clipping of audiosignals

Seznam použitých zkratek a symbolů

α, β, γ, φ	–	Skaláry nebo skalární funkce (dle kontextu)
$\mathbf{x}$, $\mathbf{b}$, $\mathbf{g}$	–	Vektory
$\mathbf{A}$	–	Matice
f, g	–	Zobrazení z $\mathbb{R}^n$ do $\mathbb{R}$
$\boldsymbol{\beta}$, $\boldsymbol{\varphi}$	–	Zobrazení z $\mathbb{R}^n$ do $\mathbb{R}^n$
Ω, Ω_B	–	Množiny
k	–	Čítač
P_{Ω_B}	–	Ortogonální projekce na množinu Ω_B
$\|\mathbf{x}\|$	–	Eukleidovská norma vektoru $\mathbf{x}$
$(\mathbf{x}, \mathbf{y})$	–	Eukleidovský skalární součin vektorů $\mathbf{x}$ a $\mathbf{y}$
$\kappa(\mathbf{A})$	–	Spektrální číslo podmíněnosti matice $\mathbf{A}$ (Je rovno podílu největšího a nejmenšího vlastního čísla matice)
KKT	–	Karush-Kuhn-Tuckerovy podmínky
MPRGP	–	Modified proportioning with reduced gradient projections (Metoda modifikované proporcionalizace s redukovanými projekcemi gradientu

Obsah

1 Úvod

Optimalizace je snaha o nalezení nejlepšího prvku z určíté množiny pro zadané kritéria. V nejjednodušším případě se jedná o minimalizaci nebo maximalizaci reálné funkce. Mnoho teoretických i reálných úloh vede na řešení těchto optimalizačních úloh. Cílem této práce je popsat vybrané algoritmy určené k řešení minimalizační úlohy a aplikovat je pro úlohy ořezání audiosignálu a modelování průhybu struny.

Nejdříve si v druhé kapitole zformulujeme minimalizační úlohu bez omezení a uvedeme si metodu největšího spádu, metodu sdružených gradientů a Barzilai-Borweinův algoritmus používáné pro její řešení. Ve třetí kapitole se budeme zabývat úlohou s lineárními nerovnostními omezeními a předvedeme si MPRGP algoritmus vyvinutý prof. Dostálem. Ve čtvrté kapitole porovnáme uvedené algoritmy na úlohách modelování průhybu struny a ořezání zvukového signálu.

2 Gradientní metody

V této kapitole si představíme některé gradientní metody detailně popsané v [1] a [3]. Tyto metody jsou určené k řešení minimalizační úlohy

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \min_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}(\mathbf{A}\mathbf{x},\mathbf{x}) - (\mathbf{b},\mathbf{x}),$$

kde

$$f : \mathbb{R}^n \to \mathbb{R}, \mathbf{A} \in \mathbb{R}^{n\times n}, \mathbf{b} \in \mathbb{R}^n$$

a kde matice $\mathbf{A}$ je symetrická, pozitivně definitní.

Lemma 2.1 *Řešení minimalizační úlohy je ekvivalentní s řešením soustavy* $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Důkaz. Viz [1, strana 54]. ∎

2.1 Metoda největšího spádu

Metoda největšího spádu v každém kroku k minimalizuje funkci $f(\mathbf{x})$ ve směru vektoru $\mathbf{v}_k$ s délkou kroku α_k. Aproximaci řešení v dalším kroku $k+1$ proto získáme z předpisu

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \tag{1}$$

kde

$$\mathbf{x}_k \in \mathbb{R}^n, \mathbf{v}_k \in \mathbb{R}^n, \alpha_k \in \mathbb{R}.$$

Volba směru minimalizace

Jako vhodný směr pro minimalizaci funkce se zdá směr, ve kterém funkce nejvíce klesá, tj. směr největšího spádu. Derivace funkce f ve směru $\mathbf{v}_k$ by proto měla být co nejmenší

$$\frac{\mathrm{d}f(\mathbf{x}_k)}{\mathrm{d}\mathbf{v}_k} = (\nabla f(\mathbf{x}_k), \mathbf{v}_k) = \cos\varphi \cdot \|\mathbf{g}_k\| \cdot \|\mathbf{v}_k\|.$$

Nejmenší bude tato hodnota, pokud zvolíme $\cos\varphi = -1 \Leftrightarrow \varphi = \pi$. Úhel mezi vektorem $\mathbf{v}_k$ a vektorem $\mathbf{g}_k$ je π, proto $\mathbf{v}_k = -\mathbf{g}_k$. Vhodným směrem pro minimalizaci funkce je tedy směr opačného gradientu.

Délka kroku

Zbývá zvolit optimální délku kroku α_k, než začne funkce ve směru $\mathbf{v}_k$ znovu stoupat. Zvolíme si funkci $\varphi(\alpha) = f(\mathbf{x}_k + \alpha\mathbf{v}_k)$ reálné proměnné α. Po úpravách dostáváme

$$\begin{aligned}
\varphi(\alpha) &= \tfrac{1}{2}(\mathbf{A}(\mathbf{x}_k + \alpha\mathbf{v}_k), \mathbf{x}_k + \alpha\mathbf{v}_k) - (\mathbf{b}, \mathbf{x}_k + \alpha\mathbf{v}_k) \\
&= \tfrac{1}{2}(\mathbf{A}\mathbf{x}_k + \alpha\mathbf{A}\mathbf{v}_k, \mathbf{x}_k + \alpha\mathbf{v}_k) - (\mathbf{b}, \mathbf{x}_k) - (\mathbf{b}, \alpha\mathbf{v}_k) \\
&= \tfrac{1}{2}(\mathbf{A}\mathbf{x}_k, \mathbf{x}_k) + \tfrac{1}{2}(\mathbf{A}\mathbf{x}_k, \alpha\mathbf{v}_k) + \tfrac{1}{2}(\alpha\mathbf{A}\mathbf{v}_k, \mathbf{x}_k) \\
&\quad + \tfrac{1}{2}(\alpha\mathbf{A}\mathbf{v}_k, \alpha\mathbf{v}_k) - (\mathbf{b}, \mathbf{x}_k) - (\mathbf{b}, \alpha\mathbf{v}_k) \\
&= \tfrac{1}{2}(\mathbf{A}\mathbf{x}_k, \mathbf{x}_k) - (\mathbf{b}, \mathbf{x}_k) + \tfrac{1}{2}(\mathbf{A}\alpha\mathbf{v}_k, \alpha\mathbf{v}_k) + (\alpha\mathbf{A}\mathbf{x}_k, \mathbf{v}_k) - (\mathbf{b}, \alpha\mathbf{v}_k) \\
&= f(\mathbf{x}_k) + \tfrac{1}{2}(\mathbf{A}\alpha\mathbf{v}_k, \alpha\mathbf{v}_k) + \alpha((\mathbf{A}\mathbf{x}_k, \mathbf{v}_k) - (\mathbf{b}, \mathbf{v}_k)) \\
&= f(\mathbf{x}_k) + \tfrac{1}{2}(\mathbf{A}\alpha\mathbf{v}_k, \alpha\mathbf{v}_k) + \alpha(\mathbf{A}\mathbf{x}_k - \mathbf{b}, \mathbf{v}_k) \\
&= f(\mathbf{x}_k) + \tfrac{1}{2}\alpha^2(\mathbf{A}\mathbf{v}_k, \mathbf{v}_k) + \alpha(\mathbf{g}_k, \mathbf{v}_k).
\end{aligned} \tag{2}$$

Hledáme takové α, aby byla funkce $\varphi(\alpha)$ minimální. Proto položíme derivaci funkce $\varphi(\alpha)$ rovnu 0 a vyřešíme rovnici

$$\varphi'(\alpha) = \alpha(\mathbf{A}\mathbf{v}_k, \mathbf{v}_k) + (\mathbf{g}_k, \mathbf{v}_k) = 0.$$

Získáváme stacionární bod

$$\alpha_k = -\frac{(\mathbf{g}_k, \mathbf{v}_k)}{(\mathbf{A}\mathbf{v}_k, \mathbf{v}_k)}.$$

Protože je matice $\mathbf{A}$ pozitivně definitní, tak

$$\varphi''(\alpha) = (\mathbf{A}\mathbf{v}_k, \mathbf{v}_k) > 0.$$

Z toho plyne, že minimum funkce φ je $\alpha_k = -\frac{(\mathbf{g}_k, \mathbf{v}_k)}{(\mathbf{A}\mathbf{v}_k, \mathbf{v}_k)}$. Toto dosadíme do předpisu (1) a dostáváme

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{v}_k = \mathbf{x}_k + \frac{(\mathbf{g}_k, \mathbf{v}_k)}{(\mathbf{A}\mathbf{v}_k, \mathbf{v}_k)}\mathbf{g}_k = \mathbf{x}_k - \frac{(\mathbf{g}_k, \mathbf{g}_k)}{(\mathbf{A}\mathbf{g}_k, \mathbf{g}_k)}\mathbf{g}_k. \tag{3}$$

Algoritmus a jeho vlastnosti

Input: $\mathbf{x}_0$, $\mathbf{A}$, $\mathbf{b}$, ϵ

$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$k := 0$
while $\|\mathbf{g}_k\| \geq \epsilon\|\mathbf{b}\|$ do
    $\alpha_k := \|\mathbf{g}_k\|^2/(\mathbf{A}\mathbf{g}_k, \mathbf{g}_k)$
    $\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_k\mathbf{g}_k$
    $\mathbf{g}_{k+1} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
    $k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 1: Metoda největšího spádu

Poznámka 2.1 Vztah pro výpočet gradientu $\mathbf{g}_{k+1}$ můžeme upravit

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k - \alpha_k\mathbf{g}_k) - \mathbf{b} = \mathbf{A}\mathbf{x}_k - \mathbf{A}\alpha_k\mathbf{g}_k - \mathbf{b} = \mathbf{g}_k - \alpha_k\mathbf{A}\mathbf{g}_k,$$

což je výpočetně výhodnější, protože $\mathbf{A}\mathbf{g}_k$ už známe z výpočtu α_k.

Lemma 2.2 *Pro libovolnou počáteční aproximaci $\mathbf{x}_0 \in \mathbb{R}^n$ konverguje metoda největšího spádu k řešení rovnice $\mathbf{A}\mathbf{x} = \mathbf{b}$. Pro chybu dvou po sobě následujících iterací navíc platí*

$$(\mathbf{A}\mathbf{e}_{k+1}, \mathbf{e}_{k+1}) \leq (1 - \frac{1}{\kappa(\mathbf{A})})(\mathbf{A}\mathbf{e}_k, \mathbf{e}_k),$$

kde $\mathbf{e}_k$ je chyba k-té iterace metody největšího spádu a $\kappa(\mathbf{A})$ je spektrální číslo podmíněnosti matice $\mathbf{A}$.

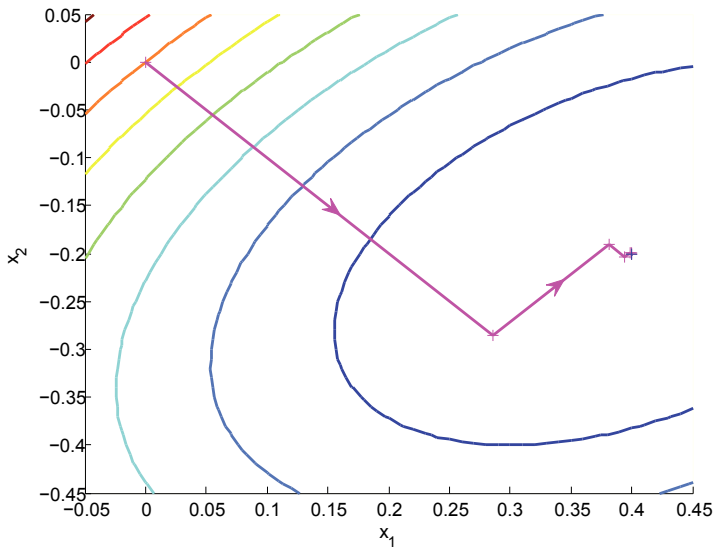
Důkaz. Viz [1, strana 57]. ∎

Uvedeme si jednoduchý příklad, na kterém si ukážeme postup iterací metody největšího spádu.

Příklad 2.1
Nalezněte řešení soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Zvolme počáteční aproximaci $\mathbf{x}_0 = (0, 0)^T$ a požadovanou přesnost řešení $\epsilon = 10^{-4}$. Potom algoritmus metody největšího spádu nalezne řešení $\bar{\mathbf{x}} = (0.4, -0.2)^T$ s požadovanou přesností po 7 iteracích, které ukazuje obrázek 1. ∎



Obrázek 1: Iterace metody největšího spádu

2.2 Metoda sdružených gradientů

Pro minimalizaci funkce $f(\mathbf{x})$ existuje i efektivnější metoda, než metoda největšího spádu. Touto metodou je metoda sdružených gradientů, která používá jako směr minimalizace sdružené směry, které konvergují k řešení mnohem rychleji, než směry největšího spádu.

Sdružené směry

Než se dostaneme k samotné metodě, tak si zadefinujeme pojem sdružené směry a uvedeme některé jejich vlastnosti.

Definice 2.1 *Mějme neprázdnou množinu nenulových vektorů* $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\}$. *Tato množina je množinou* sdružených vektorů, *jestliže*

$$\forall i, j = 0, 1, \ldots, k : i \neq j \Rightarrow \mathbf{p}_i^T \mathbf{A} \mathbf{p}_j = 0,$$

kde $\mathbf{A}$ *je pozitivně definitní, symetrická matice.*

Poznámka 2.2 Pro definici množiny sdružených vektorů můžeme také použít skalární součin

$$\forall i, j = 0, 1, \ldots, k : i \neq j \Rightarrow (\mathbf{A}\mathbf{p}_i, \mathbf{p}_j) = 0.$$

Lemma 2.3 *Jednotlivé vektory z množiny sdružených vektorů jsou navzájem lineárně nezávislé.*

Důkaz. Viz [1, strana 61]. ∎

Předpis metody

Už víme, co to sdružené směry jsou, můžeme tedy přejít k samotnému předpisu metody sdružených gradientů.

Uvažujme v k-té iteraci množinu sdružených vektorů $\mathcal{P} := \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\}$. Další iteraci potom získáme z předpisu

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k. \tag{4}$$

Stejně jako u metody největšího spádu se snažíme zvolit koeficient α_k tak, abychom co nejvíce minimalizovali funkci f. Proto znovu zvolíme funkci $\varphi(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{p}_k)$ o jedné reálné proměnné α a budeme ji minimalizovat. Po úpravách (viz (2)) dostáváme

$$\alpha_k = -\frac{(\mathbf{p}_k, \mathbf{g}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)}.$$

Lemma 2.4 *Posloupnost $\{\mathbf{x}_k\}$ z předpisu (4) dosáhne pro jakoukoliv počáteční volbu $\mathbf{x}_0 \in \mathbb{R}^n$ řešení soustavy lineárních rovnic $\bar{\mathbf{x}} \in \mathbb{R}^n$ po maximálně n krocích.*

Důkaz. Viz [1, strana 62]. ∎

Grammův-Schmidtův A-ortogonalizační proces

Nyní potřebujeme sdružené vektory generovat. K tomu se dá využít tzv. Grammův-Schmidtův A-ortogonalizační proces (viz [2]).

Zvolme za první A-ortogonální vektor $\mathbf{p}_0$ gradient v bodě $\mathbf{x}_0$. Tento gradient není nulový, protože jinak by byl bod $\mathbf{x}_0$ řešením.

Máme množinu sdružených vektorů $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\}$ a potřebujeme najít vektor $\mathbf{p}_{k+1}$ takový, aby splňoval

$$\forall i = 0, 1, \ldots, k : (\mathbf{A}\mathbf{p}_{k+1}, \mathbf{p}_i) = 0. \tag{5}$$

Proto vyjádříme vektor $\mathbf{p}_{k+1}$ jako

$$\mathbf{p}_{k+1} = \mathbf{g}_{k+1} - \beta_{k+1,0}\mathbf{p}_0 - \beta_{k+1,1}\mathbf{p}_1 - \ldots - \beta_{k+1,k}\mathbf{p}_k = \mathbf{g}_{k+1} - (\sum_{j=0}^{k} \beta_{k+1,j}\mathbf{p}_j). \tag{6}$$

Toto dosadíme do (5) a dostaneme

$$0 = (\mathbf{A}\mathbf{p}_{k+1}, \mathbf{p}_i) = (\mathbf{A}(\mathbf{g}_{k+1} - \sum_{j=0}^{k} \beta_{k+1,j}\mathbf{p}_j), \mathbf{p}_i) = (\mathbf{A}\mathbf{g}_{k+1}, \mathbf{p}_i) - \sum_{j=0}^{k} \beta_{k+1,j}(\mathbf{A}\mathbf{p}_j, \mathbf{p}_i).$$

Celou rovnici můžeme zjednodušit díky A-ortogonalitě vektorů $\mathbf{p}_i$, kde $i = 0, \ldots, k$. Dostaneme tvar

$$0 = (\mathbf{A}\mathbf{g}_{k+1}, \mathbf{p}_i) - \beta_{k+1,i}(\mathbf{A}\mathbf{p}_i, \mathbf{p}_i).$$

Z čehož můžeme vyjádřit

$$\beta_{k+1,i} = \frac{(\mathbf{A}\mathbf{g}_{k+1}, \mathbf{p}_i)}{(\mathbf{A}\mathbf{p}_i, \mathbf{p}_i)} = \frac{(\mathbf{A}\mathbf{p}_i, \mathbf{g}_{k+1})}{(\mathbf{A}\mathbf{p}_i, \mathbf{p}_i)}.$$

Dosazením do (6) dostaneme předpis pro výpočet dalšího sdruženého vektoru

$$\mathbf{p}_{k+1} = \mathbf{g}_{k+1} - \sum_{j=0}^{k} \frac{(\mathbf{A}\mathbf{p}_j, \mathbf{g}_{k+1})}{(\mathbf{A}\mathbf{p}_j, \mathbf{p}_j)} \mathbf{p}_j. \tag{7}$$

Tento předpis ale není příliš vhodný, protože by bylo nutné pamatovat si všechny předchozí směry a v každém kroku je znovu procházet. Naštěstí se dá tento předpis výrazně zjednodušit.

Lemma 2.5 *Mějme množinu gradientů* $\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k+1}\}$*, z které pomocí předpisu (7) dostaneme množinu sdružených vektorů* $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_k\}$*. Potom platí*

$$(\mathbf{A}\mathbf{p}_j, \mathbf{g}_{k+1}) = \begin{cases} = 0 & \text{pro } j < k \\ \neq 0 & \text{pro } j = k \end{cases}$$

Důkaz. Viz [1, strana 66]. ∎

Poznámka 2.3 Z předchozí věty vyplývá

$$\beta_{k+1,i} = \frac{(\mathbf{A}\mathbf{p}_i, \mathbf{g}_{k+1})}{(\mathbf{A}\mathbf{p}_i, \mathbf{p}_i)} = \begin{cases} = 0 & \text{pro } i < k, \\ \neq 0 & \text{pro } i = k. \end{cases}$$

Což znamená, že jediný koeficient $\beta_{k+1,k}$ není nulový, proto

$$\beta_{k+1} = \beta_{k+1,k} = \frac{(\mathbf{A}\mathbf{p}_k, \mathbf{g}_{k+1})}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)}.$$

Předpis (7) se tedy dá zjednodušit na

$$\mathbf{p}_{k+1} = \mathbf{g}_{k+1} - \beta_{k+1}\mathbf{p}_k = \mathbf{g}_{k+1} - \frac{(\mathbf{A}\mathbf{p}_k, \mathbf{g}_{k+1})}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)} \mathbf{p}_k. \tag{8}$$

Tento předpis pro výpočet dalšího sdruženého směru používá jenom sdružený směr z předchozí iterace a je proto mnohem méně výpočetně i paměťově náročný.

Algoritmus a jeho vlastnosti

Input: $\mathbf{x}_0$, $\mathbf{A}$, $\mathbf{b}$, ϵ

$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$\mathbf{p}_0 := \mathbf{g}_0$
$k := 0$
while $\|\mathbf{g}_k\| \geq \epsilon\|\mathbf{b}\|$ do
  $\alpha_k := -(\mathbf{p}_k, \mathbf{g}_k)/(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)$
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
  $\mathbf{g}_{k+1} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
  $\beta_{k+1} := (\mathbf{A}\mathbf{p}_k, \mathbf{g}_{k+1})/(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)$
  $\mathbf{p}_{k+1} := \mathbf{g}_{k+1} - \beta_{k+1}\mathbf{p}_k$
  $k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 2: Metoda sdružených gradientů

Poznámka 2.4 V algoritmu můžeme upravit vztah pro výpočet gradientu $\mathbf{g}_{k+1}$

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k + \alpha_k\mathbf{p}_k) - \mathbf{b} = \mathbf{A}\mathbf{x}_k + \mathbf{A}\alpha_k\mathbf{p}_k - \mathbf{b} = \mathbf{g}_k + \alpha_k\mathbf{A}\mathbf{p}_k.$$

Toto je výpočetně výhodnější, protože $\mathbf{A}\mathbf{p}_k$ už známe z výpočtu α_k.

Lemma 2.6 *V metodě sdružených gradientů můžeme koeficienty α_k a β_{k+1} vyjádřit rovněž ve tvaru*

$$\alpha_k = \frac{\|\mathbf{g}_k\|^2}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}, \qquad \beta_{k+1} = -\frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2}.$$

Důkaz. Viz [1, strana 69]. ∎

Poznámka 2.5

1. Hledání řešení algoritmus ukončí při určité přesnosti, proto lze metodu sdružených gradientů považovat za iterační metodu.
2. Řešení $\bar{\mathbf{x}}$ soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$ náleží do tzv. Krylových prostorů K_n, které můžeme popsat pomocí lineárního obalu množiny vektorů
$$\bar{\mathbf{x}} \in K_n = \langle \mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_{n-1}\rangle = \langle \mathbf{g}_0, \mathbf{A}\mathbf{g}_0, \mathbf{A}^{n-1}\mathbf{g}_0\rangle.$$
3. Metoda sdružených gradientů nalezne řešení po n krocích. Pokud ale nastane situace, kdy řešení
$$\bar{\mathbf{x}} \in \langle \mathbf{g}_0, \mathbf{A}\mathbf{g}_0, \mathbf{A}^{m-1}\mathbf{g}_0\rangle$$
a tedy gradient $\mathbf{g}_m = 0$, potom algoritmus nalezne řešení po $m < n$ krocích.

Lemma 2.7 *Pro chybu dvou následujících iterací metody sdružených gradientů platí*

$$(\mathbf{A}\mathbf{e}_{k+1}, \mathbf{e}_{k+1}) \leq \left(\frac{\kappa(\mathbf{A}) - 1}{\kappa(\mathbf{A}) + 1}\right)^2 (\mathbf{A}\mathbf{e}_k, \mathbf{e}_k),$$

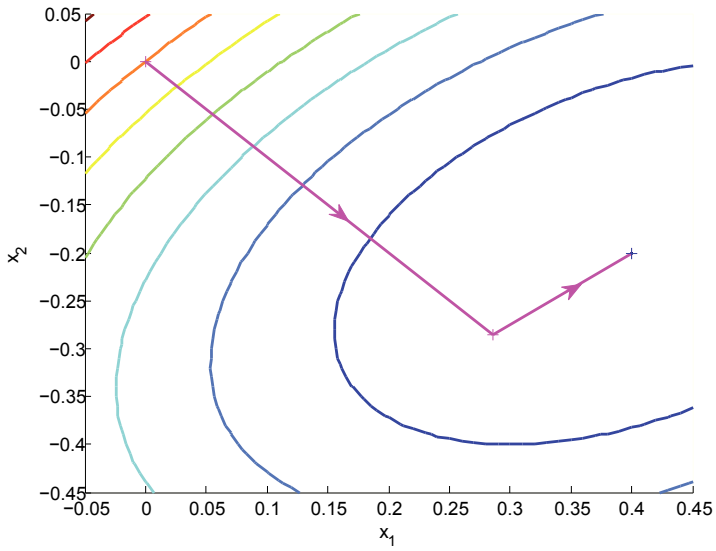
Důkaz. Viz [1, strana 70]. ∎

Postup iterací metody sdružených gradientů si ukážeme na stejném příkladu jako u metody největšího spádu (příklad 2.1).

Příklad 2.2
Nalezněte řešení soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Znovu zvolíme počáteční aproximaci $\mathbf{x}_0 = (0, 0)^T$ a požadovanou přesnost řešení $\epsilon = 10^{-4}$. Metoda sdružených gradientů nalezne řešení $\bar{\mathbf{x}} = (0.4, -0.2)^T$ dle očekávání po 2 iteracích. Postup iterací algoritmu vidíme na obrázku 2. ∎



Obrázek 2: Iterace metody sdružených gradientů

2.3 Barzilai-Borweinův algoritmus

Barzilai-Borweinův algoritmus stejně jako metoda největšího spádu minimalizuje funci f ve směru opačného gradientu. Pro výpočet nového kroku tedy platí

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k. \tag{9}$$

Rozdíl mezi metodou největšího spádu a Barzilai-Borweinovým algoritmem je ve volbě kroku α_k. Tato změna může výrazně snížit počet iterací potřebných pro nalezení řešení $\bar{\mathbf{x}}$.

Délka kroku

Při určení správného koeficientu α vyjdeme z Newtonovy metody [1] pro řešení rovnice $\mathbf{g}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{o}$, která má předpis (pro $n = 1$)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}(\mathbf{x}_k)}{\mathbf{g}'(\mathbf{x}_k)}, \tag{10}$$

kde derivaci funkce $\mathbf{g}$ můžeme aproximovat výrazem

$$\mathbf{g}'(\mathbf{x}_k) \approx \frac{\mathbf{g}(\mathbf{x}_k) - \mathbf{g}(\mathbf{x}_{k-1})}{\mathbf{x}_k - \mathbf{x}_{k-1}}.$$

Po dosazení do (10) dostáváme

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\mathbf{g}(\mathbf{x}_k) - \mathbf{g}(\mathbf{x}_{k-1})}\mathbf{g}(\mathbf{x}_k).$$

Označme $\mathbf{g}(\mathbf{x}_k)$ jako gradient $\mathbf{g}_k$ funkce f a výraz

$$\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\mathbf{g}(\mathbf{x}_k) - \mathbf{g}(\mathbf{x}_{k-1})}$$

jako délku kroku α_k. Vznikne nám předpis

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k,$$

který je totožný s (9). Pro výpočet koeficientu α_k proto platí

$$\alpha_k = \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\mathbf{g}_k - \mathbf{g}_{k-1}}.$$

Jelikož tento výraz nelze vyhodnotit pro více dimenzí ($n > 1$), musíme místo toho vyřešit rovnici

$$\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\alpha} = \mathbf{g}_k - \mathbf{g}_{k-1}$$

metodou nejmenších čtverců [1]. Dostaneme

$$\begin{aligned} \frac{1}{\alpha_k} &= \arg\min_{\alpha\in\mathbb{R}} \|(\mathbf{x}_k - \mathbf{x}_{k-1})\frac{1}{\alpha} - (\mathbf{g}_k - \mathbf{g}_{k-1})\|^2 \\ &= \arg\min_{\alpha\in\mathbb{R}} \frac{1}{2}\|(\mathbf{x}_k - \mathbf{x}_{k-1})\frac{1}{\alpha} - (\mathbf{g}_k - \mathbf{g}_{k-1})\|^2. \end{aligned} \tag{11}$$

Označíme si

$$\begin{aligned} \mathbf{u}_k &= \mathbf{x}_k - \mathbf{x}_{k-1}, \\ \mathbf{w}_k &= \mathbf{g}_k - \mathbf{g}_{k-1}, \end{aligned} \tag{12}$$

potom upravením (11) dostaneme

$$\frac{1}{\alpha_k} = \arg\min_{\alpha\in\mathbb{R}}[\frac{1}{2\alpha^2}(\mathbf{u}_k,\mathbf{u}_k) - \frac{1}{\alpha}(\mathbf{u}_k,\mathbf{w}_k) + \frac{1}{2}(\mathbf{w}_k,\mathbf{w}_k)].$$

Zvolme si funkci

$$h(\alpha) = \frac{1}{2\alpha^2}(\mathbf{u}_k,\mathbf{u}_k) - \frac{1}{\alpha}(\mathbf{u}_k,\mathbf{w}_k) + \frac{1}{2}(\mathbf{w}_k,\mathbf{w}_k).$$

Z nutné podmínky existence minima [4] platí

$$h'(\alpha) = -\frac{1}{\alpha^3}(\mathbf{u}_k,\mathbf{u}_k) + \frac{1}{\alpha^2}(\mathbf{u}_k,\mathbf{w}_k) = 0 = -\frac{1}{\alpha}(\mathbf{u}_k,\mathbf{u}_k) + (\mathbf{u}_k,\mathbf{w}_k),$$

z čehož dostáváme

$$\alpha_k = \frac{(\mathbf{u}_k,\mathbf{u}_k)}{(\mathbf{u}_k,\mathbf{w}_k)}. \tag{13}$$

Vektory (12) můžeme zapsat ve tvaru

$$\begin{aligned}\mathbf{w}_k = \mathbf{g}_k - \mathbf{g}_{k-1} = &\quad (\mathbf{A}\mathbf{x}_k - \mathbf{b}) - (\mathbf{A}\mathbf{x}_{k-1} - \mathbf{b}) &= \mathbf{A}\mathbf{u}_k,\\ \mathbf{u}_k = \mathbf{x}_k - \mathbf{x}_{k-1} = &\quad (\mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{g}_{k-1}) - \mathbf{x}_{k-1} &= \alpha_{k-1}\mathbf{g}_{k-1}.\end{aligned}$$

Dosazením do (13) dostaneme výsledný krok α_k:

$$\alpha_k = \frac{(\mathbf{u}_k,\mathbf{u}_k)}{(\mathbf{u}_k,\mathbf{A}\mathbf{u}_k)} = \frac{(\alpha_{k-1}\mathbf{g}_{k-1},\alpha_{k-1}\mathbf{g}_{k-1})}{(\alpha_{k-1}\mathbf{g}_{k-1},\alpha_{k-1}\mathbf{A}\mathbf{g}_{k-1})} = \frac{(\mathbf{g}_{k-1},\mathbf{g}_{k-1})}{(\mathbf{g}_{k-1},\mathbf{A}\mathbf{g}_{k-1})},$$

který dosadíme do předpisu (9) a vznikne nám výsledný tvar

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{(\mathbf{g}_{k-1},\mathbf{g}_{k-1})}{(\mathbf{g}_{k-1},\mathbf{A}\mathbf{g}_{k-1})}\mathbf{g}_k. \tag{14}$$

Porovnáním s metodou největšího spádu (3) zjistíme, že Barzilai-Borweinův algoritmus používá pro délku kroku α_k předchozí krok α_{k-1} metody největšího spádu.

Algoritmus a jeho vlastnosti

Input: $\mathbf{x}_0$, $\mathbf{A}$, $\mathbf{b}$, ϵ

$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$k := 0$
$\alpha_0 := \|\mathbf{g}_0\|^2/(\mathbf{A}\mathbf{g}_0,\mathbf{g}_0)$
while $\|\mathbf{g}_k\| \geq \epsilon\|\mathbf{b}\|$ do
    $\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_k\mathbf{g}_k$
    $\alpha_{k+1} := \|\mathbf{g}_k\|^2/(\mathbf{A}\mathbf{g}_k,\mathbf{g}_k)$
    $\mathbf{g}_{k+1} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
    $k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 3: Barzilai-Borweinův algoritmus

Poznámka 2.6 Stejně jako u metody největšího spádu můžeme výpočet gradientu $\mathbf{g}_{k+1}$ upravit

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k - \alpha_k \mathbf{g}_k) - \mathbf{b} = \mathbf{A}\mathbf{x}_k - \mathbf{A}\alpha_k \mathbf{g}_k - \mathbf{b} = \mathbf{g}_k - \alpha_k \mathbf{A}\mathbf{g}_k.$$

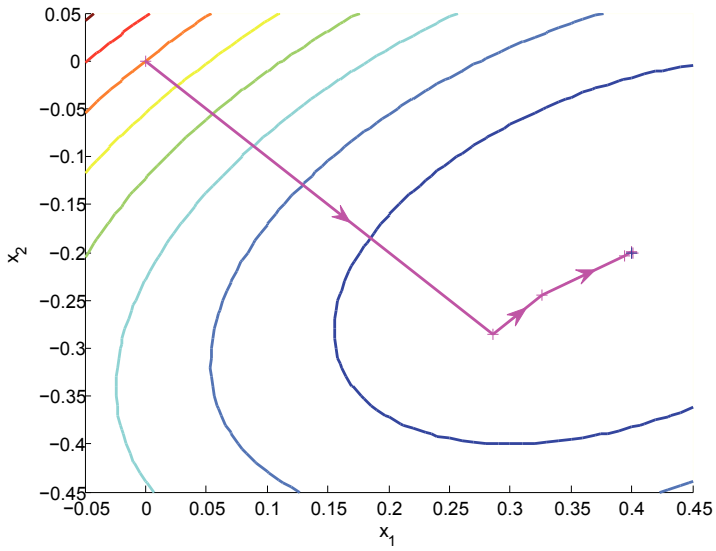
Tento vztah je výpočetně výhodnější, protože $\mathbf{A}\mathbf{g}_k$ už známe z výpočtu α_k.

Vlastnosti algoritmu si předvedeme znovu na stejném příkladu jako u předchozích metod.

Příklad 2.3
Nalezněte řešení soustavy $\mathbf{A}\mathbf{x} = \mathbf{b}$, kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Znovu zvolíme počáteční aproximaci $\mathbf{x}_0 = (0,0)^T$ a požadovanou přesnost řešení $\epsilon = 10^{-4}$. Barzilai-Borweinův algoritmus bude ukončen po 6 iteracích, což je o 1 méně než u metody největšího spádu. Počet násobení maticí $\mathbf{A}$ je ale stejný, protože Barzilai-Borweinův algoritmus násobí navíc maticí $\mathbf{A}$ pro výpočet α_0. Postupné iterace algoritmu můžeme vidět na obrázku 3. ■



Obrázek 3: Barzilai-Borweinův algoritmus - postupné iterace

3 Minimalizace s lineárními omezeními

V této kapitole se budeme zabývat minimalizační úlohou s jednoduchými lineárními omezeními $\mathbf{l} \in \mathbb{R}^n$ a $\mathbf{u} \in \mathbb{R}^n$. Budeme tedy řešit úlohu

$$\min_{\mathbf{x}\in\Omega_B} f(\mathbf{x}), \tag{15}$$

kde

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x}), \quad f : \mathbb{R}^n \to \mathbb{R}$$

a

$$\Omega_B = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}.$$

Pro vyřešení této úlohy použijeme vhodně modifikovanou metodu největšího spádu a Barzilai-Borweinův algoritmus z předchozí kapitoly a MPRGP algoritmus kombinující metody z předchozí kapitoly (viz [5]).

3.1 Metoda největšího spádu

Pokud chceme použít metodu největšího spádu pro úlohu minimalizace s omezeními musíme ji trochu upravit. Směr minimalizace zůstane stejný, tedy směr opačného gradientu, stejně jako délka kroku α_k. Úprava zpočívá v použití ortogonální projekce P_{Ω_B} na přípustnou kovnexní množinu Ω_B. Předpis pro další krok $k+1$ se tedy změní z (3) na

$$\mathbf{x}_{k+1} = P_{\Omega_B}(\mathbf{x}_k - \alpha_k \mathbf{g}_k) = P_{\Omega_B}(\mathbf{x}_k - \frac{(\mathbf{g}_k, \mathbf{g}_k)}{(\mathbf{A}\mathbf{g}_k, \mathbf{g}_k)}\mathbf{g}_k). \tag{16}$$

Zadefinujme si nyní, co to je projekce na konvexní množinu.

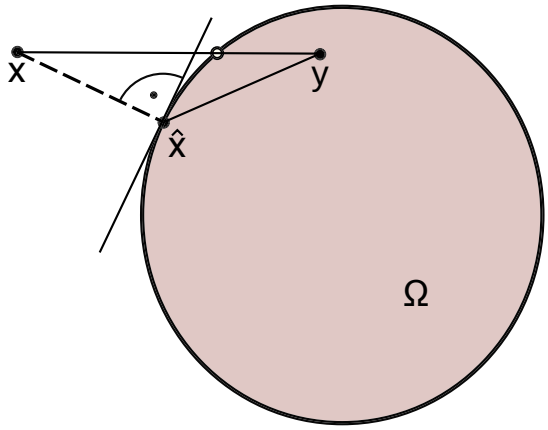
Ortogonální projekce na konvexní množinu

Ortogonální projekcí na konvexní množinu $\Omega \subset \mathbb{R}^n$ rozumíme zobrazení, které každému vektoru $\mathbf{x} \in \mathbb{R}^n$ přiřadí nejbližší vektor $\hat{\mathbf{x}} \in \Omega$ (viz obrázek 4). Vzdálenost vektorů $\mathbf{x}$ a $\hat{\mathbf{x}}$ můžeme změřit jako normu eukleidovského skalárního součinu těchto vektorů.

Tvrzení 3.1 *Nechť $\Omega \subset \mathbb{R}^n$ je neprázdná, uzavřená konvexní množina a $\mathbf{x} \in \mathbb{R}^n$. Potom existuje právě jeden bod $\hat{\mathbf{x}} \in \Omega$ s minimální eukleidovskou vzdáleností od $\mathbf{x}$. Zároveň pro každé $\mathbf{y} \in \Omega$ platí*

$$(\mathbf{x} - \hat{\mathbf{x}})^T(\mathbf{y} - \hat{\mathbf{x}}) \leq 0.$$

Důkaz. Viz [5, strana 36]. ∎



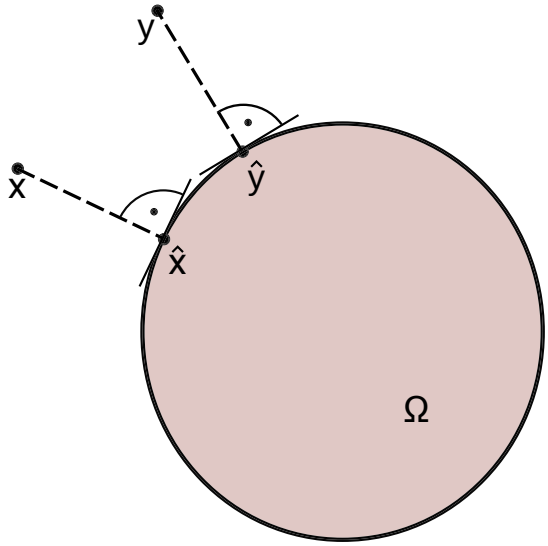
Obrázek 4: Projekce na konvexní množinu

Z tvrzení 3.1 se dá ukázat, že projekce P_Ω je neexpanzivní (viz obrázek 5).

Důsledek 3.1 *Nechť $\Omega \subset \mathbb{R}^n$ je neprázdná, uzavřená konvexní množina a pro každé $\mathbf{x} \in \mathbb{R}^n$ označíme projekci $\mathbf{x}$ na množinu Ω jako $\hat{\mathbf{x}} \in \Omega$. Pak pro každé $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ platí*

$$\|\hat{\mathbf{x}} - \hat{\mathbf{y}}\| \leq \|\mathbf{x} - \mathbf{y}\|.$$

Důkaz. Viz [5, strana 37]. ∎

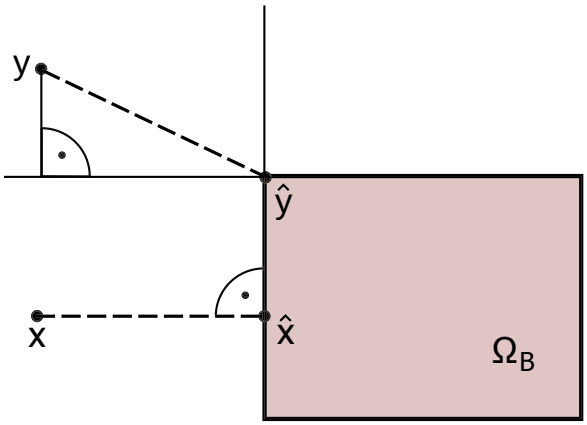


Obrázek 5: Projekce P_Ω je neexpanzivní

Poznámka 3.1 Vzhledem k tomu, že v celé kapitole uvažujeme přípustnou množinu Ω_B, můžeme projekci P_{Ω_B} zadefinovat vztahem

$$P_{\Omega_B}(\mathbf{x}) = \mathbf{y}: \qquad \mathbf{y} \in \mathbb{R}^n,\ \mathbf{y}_i = \max\{\mathbf{l}_i, \min\{\mathbf{u}_i, \mathbf{x}_i\}\},\ i = 1, \ldots, n.$$

Takto zadefinovanou projekci můžeme vidět na obrázku 6.



Obrázek 6: Projekce na množinu Ω_B

Algoritmus metody

Poznámka 3.2 Protože algoritmus metody nehledá minimum v celém prostoru, ale jen na množině Ω_B, musíme pozměnit jeho ukončovací podmínku. Místo klasického gradientu g použijeme projektovaný gradient $\mathbf{g}^P$ splňující KKT podmínky (definice viz algoritmus MPRGP).

Input: $\mathbf{A}$, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, ϵ, $\Omega_B = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$, $\mathbf{x}_0 \in \Omega_B$

$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$\mathbf{g}_0^P := \mathbf{g}_0$
$k := 0$
while $\|\mathbf{g}_k^P\| \geq \epsilon\|\mathbf{b}\|$ do
  $\alpha_k := \|\mathbf{g}_k\|^2/(\mathbf{A}\mathbf{g}_k, \mathbf{g}_k)$
  $\mathbf{x}_{k+1} := P_{\Omega_B}(\mathbf{x}_k - \alpha_k\mathbf{g}_k)$
  $\mathbf{g}_{k+1} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
  $\mathbf{g}_{k+1}^P := (\mathbf{x}_{k+1} - \mathbf{x}_k)/\alpha_k$
  $k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 4: Metoda největšího spádu s projekcí

Na jednoduchém příkladu si předvedeme fungování algoritmu.

Příklad 3.1
Nalezněte minimum funkce

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x})$$

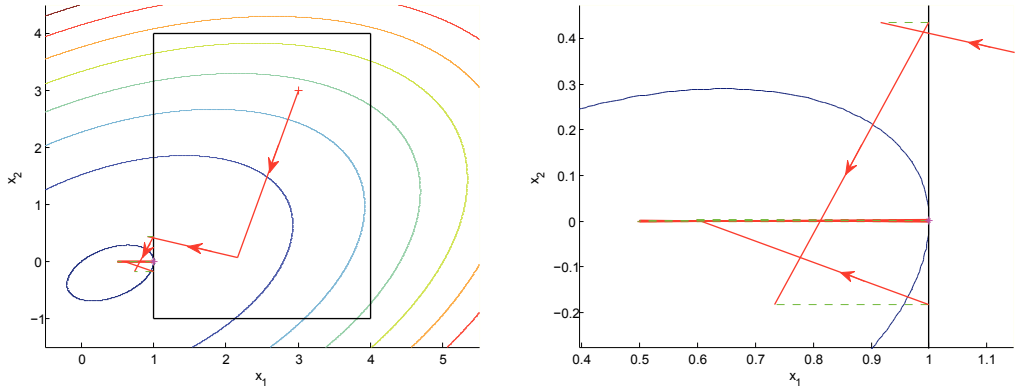
na množině

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\},$$

kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{l} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

Zvolíme-li počáteční aproximaci $\mathbf{x}_0 = (3, 3)^T$ a požadovanou přesnost řešení $\epsilon = 10^{-4}$, algoritmus nalezne řešení $\bar{\mathbf{x}} = (1, 0)^T$ s požadovanou přesností po 12 iteracích. Na obrázku 7 je vykreslen postup iterací vzhledem k vrstevnicím funkce f a množině Ω. ■



Obrázek 7: Postup iterací metody největšího spádu (vpravo přiblížení)

3.2 Barzilai-Borweinův algoritmus

Barzilai-Borweinův algoritmus se stejně jako metoda největšího spádu pro úlohu minimalizace s omezeními liší od své základní verze v použití ortogonální projekce na přípustnou množinu P_{Ω_B}. Výsledný předpis pro výpočet dalšího kroku $k+1$ se změní na

$$\mathbf{x}_{k+1} = P_{\Omega_B}(\mathbf{x}_k - \alpha_k \mathbf{g}_k) = P_{\Omega_B}(\mathbf{x}_k - \frac{(\mathbf{g}_{k-1}, \mathbf{g}_{k-1})}{(\mathbf{g}_{k-1}, \mathbf{A}\mathbf{g}_{k-1})}\mathbf{g}_k). \tag{17}$$

Stejně jako u metody největšího spádu musíme v Barzilai-Borweinově algoritmu pro ukončovací podmínku použít projektovaný gradient $\mathbf{g}^P$. Algoritmus pak vypadá následovně:

Input: $\mathbf{A}$, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, ϵ, $\Omega_B = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$, $\mathbf{x}_0 \in \Omega_B$

$\mathbf{g}_0 := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$\mathbf{g}_0^P := \mathbf{g}_0$
$k := 0$
$\alpha_0 := \|\mathbf{g}_0\|^2/(\mathbf{A}\mathbf{g}_0, \mathbf{g}_0)$
while $\|\mathbf{g}_k^P\| \geq \epsilon\|\mathbf{b}\|$ do
    $\mathbf{x}_{k+1} := P_{\Omega_B}(\mathbf{x}_k - \alpha_k\mathbf{g}_k)$
    $\alpha_{k+1} := \|\mathbf{g}_k\|^2/(\mathbf{A}\mathbf{g}_k, \mathbf{g}_k)$
    $\mathbf{g}_{k+1} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
    $\mathbf{g}_{k+1}^P := (\mathbf{x}_{k+1} - \mathbf{x}_k)/\alpha_k$
    $k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 5: Barzilai-Borweinův algoritmus s projekcí

Příklad 3.2
Nalezněte minimum funkce

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x})$$

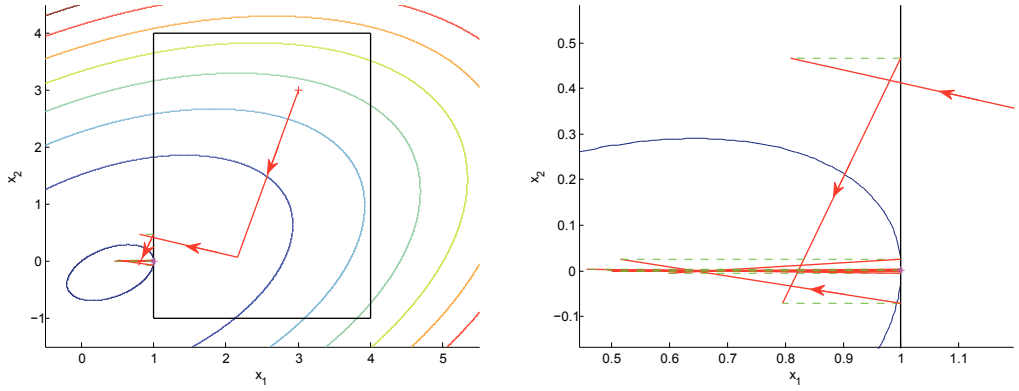
na množině

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\},$$

kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{l} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

Znovu zvolíme počáteční aproximaci $\mathbf{x}_0 = (3, 3)^T$. Barzilai-Borweinův algoritmus nalezne řešení $\bar{\mathbf{x}} = (1, 0)^T$ s požadovanou přesností $\epsilon = 10^{-4}$ po 14 iteracích. Postup iterací vzhledem k vrstevnicím funkce f a množině Ω můžeme vidět na obrázku 8. ■



Obrázek 8: Postup iterací Barzilai-Borweinova algoritmu (vpravo přiblížení)

3.3 MPRGP algoritmus

MPRGP algoritmus vychází z metody sdružených gradientů, kterou vhodně kombinuje s dalšími kroky pro splnění jednoduchých omezení. Než se ale dostaneme k samotnému algoritmu, zavedeme si následující značení.

Řešení minimalizační úlohy $\bar{\mathbf{x}}$ musí splňovat tzv. Karush-Kuhn-Tuckerovy podmínky, tedy pro $i = 1, 2, \ldots, n$ a $\bar{\mathbf{g}} = \mathbf{A}\mathbf{x} - \mathbf{b}$ musí platit

$$\begin{aligned} \bar{\mathbf{x}}_i = \mathbf{l}_i \quad &\Rightarrow \quad \bar{\mathbf{g}}_i \geq 0, \\ \bar{\mathbf{x}}_i = \mathbf{u}_i \quad &\Rightarrow \quad \bar{\mathbf{g}}_i \leq 0, \\ \mathbf{l}_i < \bar{\mathbf{x}}_i < \mathbf{u}_i \quad &\Rightarrow \quad \bar{\mathbf{g}}_i = 0. \end{aligned} \tag{18}$$

Označme $\mathcal{N}$ jako množinu všech indexů

$$\mathcal{N} = \{1, 2, \ldots, n\}.$$

Nazvěme množinu všech indexů $\mathcal{A}(\mathbf{x})$, pro které platí $\mathbf{x}_i = \mathbf{l}_i$ nebo $\mathbf{x}_i = \mathbf{u}_i$, aktivní množinou:

$$\mathcal{A}(\mathbf{x}) = \{i \in \mathcal{N} : \mathbf{x}_i = \mathbf{l}_i \vee \mathbf{x}_i = \mathbf{u}_i\}$$

a její doplněk $\mathcal{F}(\mathbf{x})$ volnou množinou:

$$\mathcal{F}(\mathbf{x}) = \{i \in \mathcal{N} : \mathbf{l}_i \neq \mathbf{x}_i \neq \mathbf{u}_i\}.$$

Definujme si volný gradient $\boldsymbol{\varphi}$

$$\boldsymbol{\varphi}_i(\mathbf{x}) = \begin{cases} = 0 & \text{pro } i \in \mathcal{A}(\mathbf{x}), \\ = \mathbf{g}_i(\mathbf{x}) & \text{pro } i \in \mathcal{F}(\mathbf{x}) \end{cases}$$

a uříznutý gradient $\boldsymbol{\beta}$

$$\boldsymbol{\beta}_i(\mathbf{x}) = \begin{cases} = \mathbf{g}_i^{\pm}(\mathbf{x}) & \text{pro } i \in \mathcal{A}(\mathbf{x}), \\ = 0 & \text{pro } i \in \mathcal{F}(\mathbf{x}), \end{cases}$$

kde

$$\mathbf{g}_i^{\pm}(\mathbf{x}) = \begin{cases} = \min\{\mathbf{g}_i, 0\} & \text{pro } \mathbf{x}_i = \mathbf{l}_i, \\ = \max\{\mathbf{g}_i, 0\} & \text{pro } \mathbf{x}_i = \mathbf{u}_i. \end{cases}$$

Definejme si taky projektovaný gradient $\mathbf{g}^P$

$$\mathbf{g}^P = \boldsymbol{\varphi} + \boldsymbol{\beta}.$$

Karush-Kuhn-Tuckerovy podmínky (18) jsou splněny právě tehdy, pokud tento projektovaný gradient je roven nule.

Popis algoritmu

MPRGP algoritmus obsahuje tři různé kroky, které různým způsobem nakládají s aktivní množinou $\mathcal{A}(\mathbf{x})$ a kritérium, podle kterého se určí, který z kroků se použije.

Jedním z kroků je krok sdružených gradientů vycházející z metody sdružených gradientů, který slouží k minimalizaci funkce f. V tomto kroku získáme další iteraci $k+1$ z předpisu

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_{cg}\mathbf{p}_k = \mathbf{x}_k - \frac{(\mathbf{p}_k, \mathbf{g}_k)}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)}\mathbf{p}_k. \tag{19}$$

Nové sdružené směry generujeme pomocí Grammova-Schmidtova A-ortogonalizačního procesu:

$$\mathbf{p}_{k+1} = \boldsymbol{\varphi}(\mathbf{x}_k) - \gamma\mathbf{p}_k = \boldsymbol{\varphi}(\mathbf{x}_k) - \frac{(\mathbf{A}\mathbf{p}_k, \boldsymbol{\varphi}(\mathbf{x}_k))}{(\mathbf{A}\mathbf{p}_k, \mathbf{p}_k)}\mathbf{p}_k. \tag{20}$$

Dalším krokem MPRGP algoritmu je expanzní krok. Tento krok stejně jako krok sdružených gradientů minimalizuje funci f a zároveň rozšiřuje aktivní množinu $\mathcal{A}(\mathbf{x})$. Expanzní krok se skládá ze dvou částí. Nejdříve se provede posun ve směru sdružených gradientů zkrácený na množinu Ω_B

$$\mathbf{x}_{k+\frac{1}{2}} = \mathbf{x}_k - \alpha_f\mathbf{p}, \tag{21}$$

kde

$$\alpha_f = \max\{\alpha : \mathbf{x}_k - \alpha\mathbf{p} \in \Omega_B\}.$$

Druhou částí expanzního kroku je projektovaný krok ve směru volného gradientu

$$\mathbf{x}_{k+1} = P_{\Omega_B}(\mathbf{x}_{k+\frac{1}{2}} - \bar{\alpha}\boldsymbol{\varphi}(\mathbf{x}_{k+\frac{1}{2}})) \tag{22}$$

o konstantní délce kroku $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}\rangle$.

Třetím krokem je proporcionální krok, jehož účelem je redukce aktivní množiny $\mathcal{A}(\mathbf{x})$. Tento krok je definován předpisem

$$\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_{cg}\boldsymbol{\beta}(\mathbf{x}_k) \tag{23}$$

s délkou kroku

$$\alpha_{cg} = \frac{(\boldsymbol{\beta}(\mathbf{x}_k), \mathbf{g}_k)}{(\mathbf{A}\boldsymbol{\beta}(\mathbf{x}_k), \boldsymbol{\beta}(\mathbf{x}_k))}$$

.

Zbývá nám rozhodnout, který z tří kroků se v dané iteraci algoritmu použije. K tomuto rozhodnutí použijeme kritérium

$$\|\boldsymbol{\beta}(\mathbf{x}_k)\|^2 \leq \gamma^2\tilde{\boldsymbol{\varphi}}(\mathbf{x}_k)^T\boldsymbol{\varphi}(\mathbf{x}_k), \tag{24}$$

kde $\tilde{\varphi}$ definujeme jako

$$\tilde{\boldsymbol{\varphi}} = \max\left\{\frac{\mathbf{x}_i - \mathbf{u}_i}{\bar{\alpha}}, \min\left\{\frac{\mathbf{x}_i - \mathbf{l}_i}{\bar{\alpha}}, \boldsymbol{\varphi}_i(\mathbf{x})\right\}\right\}$$

a $\gamma > 0$ je konstanta, která určuje, jak se bude minimalizovat funkce f, pokud se nezmenšuje aktivní množina. Nejlepší volbou je $\gamma = 1$.

Pokud je kritérium (24) splněno, zkusíme použít krok sdružených gradientů. Jestliže výsledek $\mathbf{x}_{k+1} \in \Omega_B$ použijeme jej, pokud ne, tak vypočítáme krok $\mathbf{x}_{k+1}$ pomocí expanzního kroku. Pokud kritérium (24) splněno není, použije se proporcionální krok.

Algoritmus a jeho vlastnosti

Input: $\mathbf{A}$, $\mathbf{b}$, $\mathbf{l}$, $\mathbf{u}$, ϵ, $\Omega_B = \{\mathbf{x} : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$, $\mathbf{x}_0 \in \Omega_B$

Inicializace
Zvolme: $\gamma > 0, \bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}\rangle$

$k := 0$
$\mathbf{g} := \mathbf{A}\mathbf{x}_0 - \mathbf{b}$
$\mathbf{p} := \boldsymbol{\varphi}(\mathbf{x}_0)$

while $\|\mathbf{g}^P(\mathbf{x}_k)\| > \epsilon\|\mathbf{b}\|$
if $\|\boldsymbol{\beta}(\mathbf{x}_k)\|^2 \leq \gamma^2 \tilde{\boldsymbol{\varphi}}(\mathbf{x}_k)^T \boldsymbol{\varphi}(\mathbf{x}_k)$
Zkouška kroku sdružených gradientů
$\alpha_{cg} := \mathbf{g}^T\mathbf{p}/\mathbf{p}^T\mathbf{A}\mathbf{p}$
$\mathbf{y} := \mathbf{x}_k - \alpha_{cg}\mathbf{p}$
$\alpha_f := max\{\alpha : \mathbf{x}_k - \alpha\mathbf{p} \in \Omega_B\}$
if $\alpha_{cg} \leq \alpha_f$
Krok sdružených gradientů
$\mathbf{x}_{k+1} := \mathbf{y}$
$\mathbf{g} := \mathbf{g} - \alpha_{cg}\mathbf{A}\mathbf{p}$
$\beta := \boldsymbol{\varphi}(\mathbf{y})^T\mathbf{A}\mathbf{p}/\mathbf{p}^T\mathbf{A}\mathbf{p}$
$\mathbf{p} := \boldsymbol{\varphi}(\mathbf{y}) - \beta\mathbf{p}$
else
Expanzní krok
$\mathbf{x}_{k+\frac{1}{2}} := \mathbf{x}_k - \alpha_f\mathbf{p}$
$\mathbf{g} := \mathbf{g} - \alpha_f\mathbf{A}\mathbf{p}$
$\mathbf{x}_{k+1} := P_{\Omega_B}(\mathbf{x}_{k+\frac{1}{2}} - \bar{\alpha}\boldsymbol{\varphi}(\mathbf{x}_{k+\frac{1}{2}}))$
$\mathbf{g} := \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}$
$\mathbf{p} := \boldsymbol{\varphi}(\mathbf{x}_{k+1})$
end if
else
Proporcionální krok
$\mathbf{d} := \boldsymbol{\beta}(\mathbf{x}_k)$
$\alpha_{cg} := \mathbf{g}^T\mathbf{d}/\mathbf{d}^T\mathbf{A}\mathbf{d}$
$\mathbf{x}_{k+1} := \mathbf{x}_k - \alpha_{cg}\mathbf{d}$
$\mathbf{g} := \mathbf{g} - \alpha_{cg}\mathbf{A}\mathbf{d}$
$\mathbf{p} := \boldsymbol{\varphi}(\mathbf{x}_{k+1})$
end if
$k := k + 1$
end while

Output: $\mathbf{x}_k$

Algoritmus 6: MPRGP algoritmus

Lemma 3.1 *Nechť posloupnost $\mathbf{x}_k$ je generována MPRGP algoritmem s počáteční aproximací $\mathbf{x}_0 \in \Omega_B$ a konstantami $\gamma > 0$ a $\bar{\alpha} \in (0, 2\|\mathbf{A}\|^{-1}\rangle$. Potom tato posloupnost konverguje k řešení minimalizační úlohy (15).*

Důkaz. Viz [5, strana 190]. ∎

Znovu si na jednoduchém příkladu ukážeme fungování algoritmu.

Příklad 3.3
Nalezněte minimum funkce

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x})$$

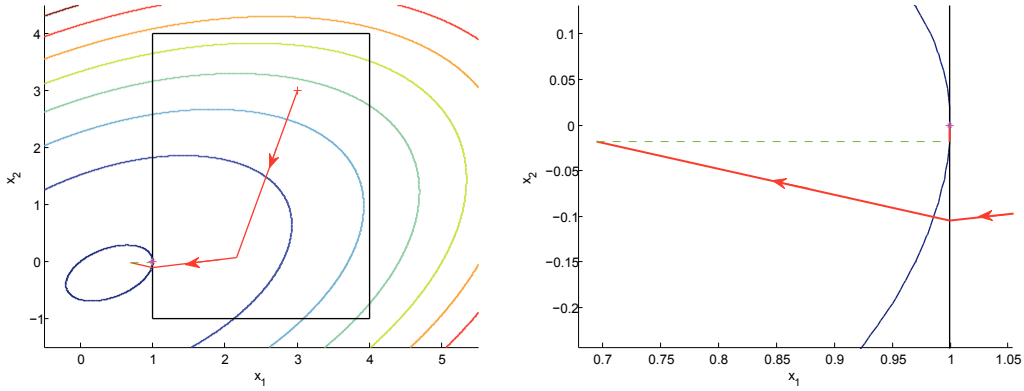
na množině

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\},$$

kde

$$\mathbf{A} = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{l} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}.$$

Pokud zvolíme počáteční aproximaci $\mathbf{x}_0 = (3, 3)^T$, $\gamma = 1$ a $\bar{\alpha} = \|\mathbf{A}\|^{-1}$, MPRGP algoritmus nalezne řešení $\bar{\mathbf{x}} = (1, 0)^T$ s požadovanou přesností $\epsilon = 10^{-4}$ už po 3 iteracích, je tedy o hodně rychlejší než předchozí algoritmy. Postup iterací vzhledem k vrstevnicím funkce f a množině Ω můžeme vidět na obrázku 9. ∎



Obrázek 9: Postup iterací MPRGP algoritmu (vpravo přiblížení)

4 Aplikace

V této kapitole si předvedeme algoritmy z předchozí kapitoly na příkladech struny a ořezání zvukového záznamu.

4.1 Struna

Uvažujme úlohu modelování průhybu u struny o délce D, upevněné na obou koncích a zatížené silou o hustotě f. Tuto úlohu můžeme vyjádřit rovnicí rovnováhy ve tvaru

$$-u'' = f \tag{25}$$

a okrajovými podmínkami $u(0) = u(D) = 0$.

K numerickému řešení této úlohy použijeme metodu sítí (viz [8]).

Z Taylorova rozvoje funkce u můžeme odvodit aproximaci druhé derivace řešení. V bodě $x + h$ je hodnota funkce u

$$u(x+h) = u(x) + u'(x)\frac{h}{1!} + u''(x)\frac{h^2}{2!} + C_1(h^3),$$

kde člen $C_1(h^3)$ můžeme pro dostatečně malé h zanedbat. Podobně pro bod $x - h$ je hodnota funkce u

$$u(x-h) = u(x) - u'(x)\frac{h}{1!} + u''(x)\frac{h^2}{2!} + C_2(h^3),$$

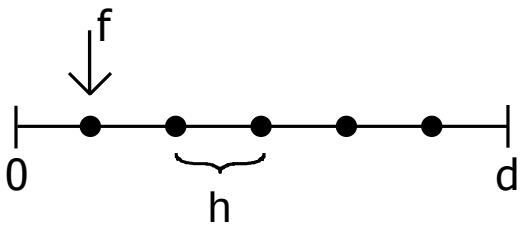
Aproximaci první derivace funkce u můžeme získat odečtením obou výrazů, pokud zanedbáme členy vyšších řádů:

$$u'(\mathbf{x}) \approx \frac{u(\mathbf{x}+h) - u(\mathbf{x}-h)}{2h}$$

Druhou derivaci naopak získáme sečtením obou výrazů

$$u''(\mathbf{x}) \approx \frac{u(\mathbf{x}-h) - 2u(\mathbf{x}) + u(\mathbf{x}+h)}{h^2} \tag{26}$$

Nyní si tuto strunu rozdělíme pravidelně s krokem h na n dílků (viz obrázek 10).



Obrázek 10: Rozdělení struny na dílky

Označíme si hodnoty řešení v uzlech $\mathbf{x}_i$ jako u_i a $f_i = f(\mathbf{x}_i)$. Pak dosazením (26) do (25) vznikne soustava lineárních rovnic

$$\begin{aligned} -(u_0 - 2u_1 + u_2) &= h^2 f_1 \\ -(u_1 - 2u_2 + u_3) &= h^2 f_2 \\ &\vdots \\ -(u_{i-1} - 2u_i + u_{i+1}) &= h^2 f_i \\ &\vdots \\ -(u_{n-2} - 2u_{n-1} + u_n) &= h^2 f_{n-1} \end{aligned} \tag{27}$$

s průhyby v krajních bodech $u_0 = u_n = 0$ a neznámými u_i, $i = 1, \ldots, n-1$, kterou můžeme zapsat v maticovém tvaru jako

$$\begin{pmatrix} 2 & -1 & 0 & & \ldots & 0 \\ -1 & 2 & -1 & 0 & & \vdots \\ 0 & -1 & 2 & -1 & \ddots & \\ & 0 & -1 & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & 2 & -1 \\ 0 & \ldots & & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{n-2} \\ u_{n-1} \end{pmatrix} = \begin{pmatrix} h^2 f_1 + u_0 \\ h^2 f_2 \\ h^2 f_3 \\ \vdots \\ h^2 f_{n-2} \\ h^2 f_{n-1} + u_n \end{pmatrix}.$$

Tuto soustavu, označenou jako $\mathbf{Au} = \mathbf{f}$, již můžeme řešit gradientními metodami popsanými v předchozích kapitolách.

4.1.1 Struna s jedním stupněm volnosti

V tomto příkladu uvažujeme strunu, která je pohyblivá jen ve směru osy y a je zatěžována silou

$$f = -30h^2.$$

Zvolme omezení

$$\begin{aligned} \mathbf{l} &= 0.5 \sin(4\pi \mathbf{x} - \frac{\pi}{3}) - 2, \\ \mathbf{u} &= (1, 1, \ldots, 1)^T, \end{aligned}$$

a požadovanou přesnost řešení $\epsilon = 10^{-2}$. Úlohu provedeme pro dimenze

$$N = n - 1 = 100, 200, 500, 1000, 2000.$$

Dosažené výsledky jsou zobrazeny na obrázku 11 a v tabulkách 1 a 2. V tabulce 1 jsou zapsány celkové počty násobení maticí $\mathbf{A}$ u jednotlivých algoritmů. Nejlepších výsledků

N	Metoda největšího spádu	BB algoritmus	MPRGP algoritmus
100	∞	163	167
200	∞	367	312
500	∞	139	910
1000	∞	509	2261
2000	∞	957	6055

Tabulka 1: Počet násobení maticí **A**

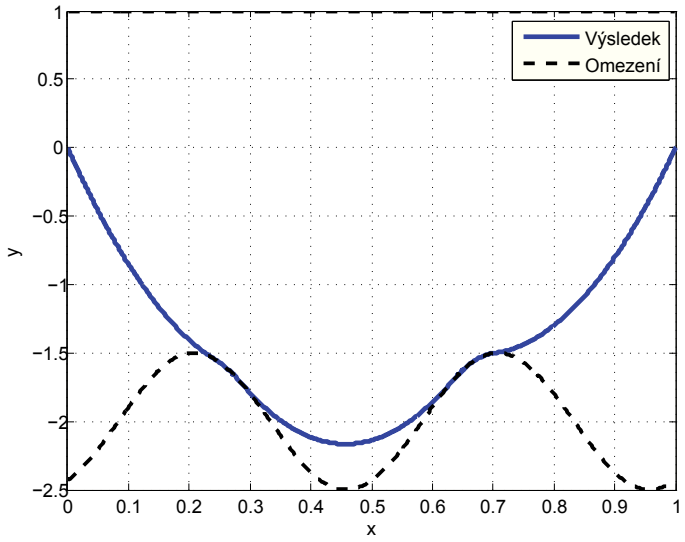
dosahuje Barzilai-Borweinův algoritmus, nejhorších metoda největšího spádu, která pro zadanou přesnost nekonvergovala k řešení v povoleném počtu iterací.

V tabulce 2 je uvedeno rozvržení provedených kroků MPRGP algoritmu a celkový počet iterací algoritmu. Nutno podotknout, že v expanzním kroku se násobí maticí **A** dvakrát, zatímco v ostatních krocích jen jednou.

N	Sdružený gradientní krok	Expanzní krok	Proporcionální krok	Celkem
100	132	16	2	150
200	256	26	3	285
500	772	66	5	843
1000	2011	120	9	2140
2000	5370	316	52	5738

Tabulka 2: MPRGP - počet jednotlivých kroků algoritmu

Na obrázku 11 pak vidíme průhyb struny vzhledem k zadaným omezením.



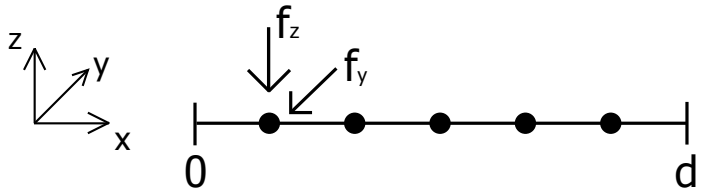
Obrázek 11: Průhyb struny

4.1.2 Struna se dvěma stupni volnosti

Uvažujme nyní strunu, která je pohyblivá nejen ve směru osy y, ale i ve směru osy z (viz obr. 12). Předpokládejme, že se tyto dva průhyby navzájem neovlivňují. Pak můžeme rovnici rovnováhy zapsat ve tvaru

$$\begin{aligned} -u_y'' &= f_y \\ -u_z'' &= f_z \end{aligned}$$

a pro každou složku funkce u platí stejné odvození jako pro strunu s jedním stupněm volnosti.

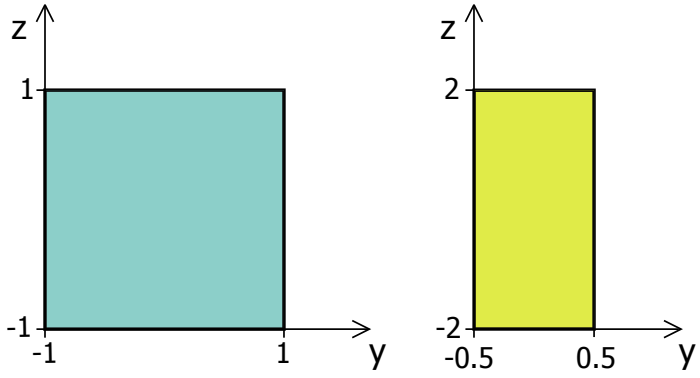


Obrázek 12: Ilustrace struny o dvou stupních volnosti

Zvolme nyní síly

$$\begin{aligned} f_y &= 8\pi^2 \sin(2\pi \mathbf{x}), \\ f_z &= 12\pi^2 \sin(2\pi \mathbf{x} - \frac{\pi}{2}). \end{aligned}$$

Omezení rozdělíme podle obrázku 13, pro první polovinu vektoru $\mathbf{x}$ použijeme omezení znázorněné modrým čtvercem, pro druhou polovinu omezení znázorněné žlutým obdelníkem.

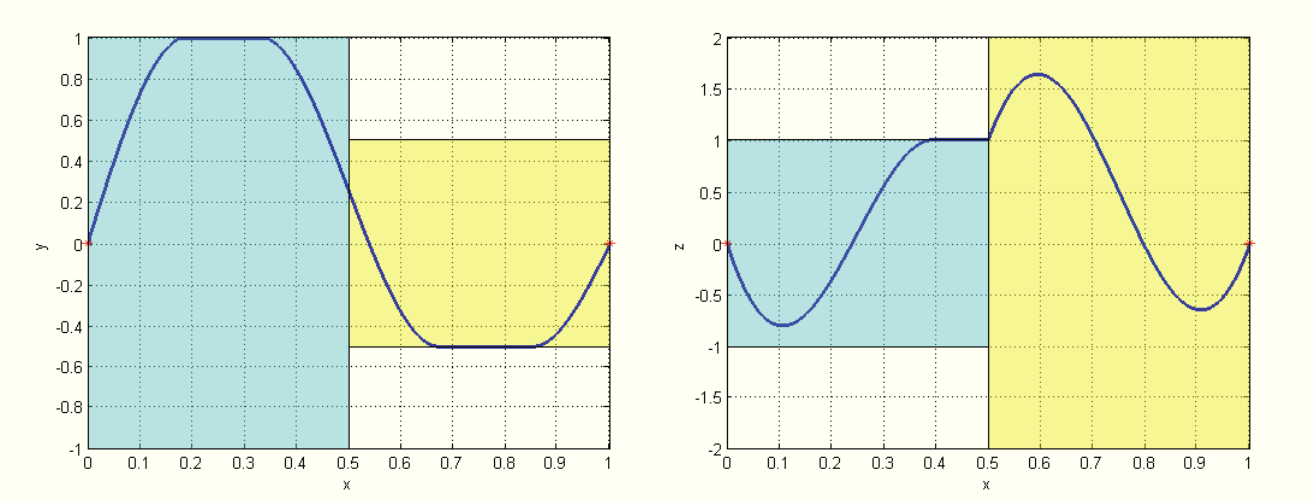


Obrázek 13: Znázornění omezení

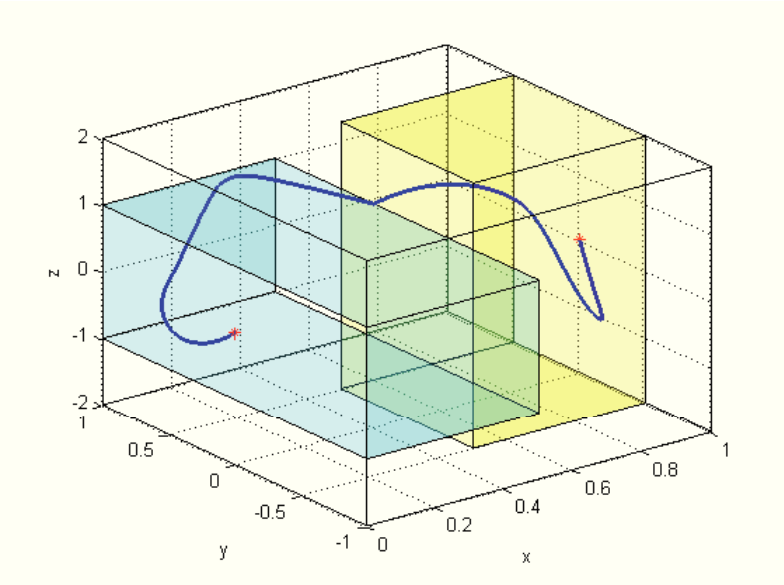
Úlohu znovu provedeme pro dimenze

$$N = 100, 200, 500, 1000, 2000$$

s požadovanou přesností řešení $\epsilon = 10^{-2}$. Výsledky algoritmů si ukážeme na obrázcích 14 a 15 a v tabulkách 3 a 4. Na obrázku 14 vidíme průhyb struny vzhledem k osám y a z, obrázek 15 pak zobrazuje průhyb struny ve 3D.



Obrázek 14: Průhyb struny ve 2D



Obrázek 15: Průhyb struny ve 3D

V tabulce 3 jsou zapsány celkové počty násobení maticí $\mathbf{A}$ u jednotlivých algoritmů. Můžeme si všimnout, že nejlepších výsledků sice dosahuje Barzilai-Borweinův algorit-

mus, ale narozdíl od algoritmu MPRGP není dostatečně stabilní a pro dimenzi $N = 100$ nekonvergoval pro požadovanou přesnost k řešení. Nejhorších výsledků dosahuje opět metoda největšího spádu.

N	Metoda největšího spádu	BB algoritmus	MPRGP algoritmus
100	∞	∞	245
200	∞	117	502
500	∞	115	1699
1000	∞	125	4433
2000	∞	259	12266

Tabulka 3: Počet násobení maticí $\mathbf{A}$

V následující tabulce je uvedeno rozvržení provedených kroků MPRGP algoritmu a celkový počet iterací algoritmu.

N	Sdružený gradientní krok	Expanzní krok	Proporcionální krok	Celkem
100	145	46	7	198
200	317	87	10	414
500	1176	246	30	1452
1000	3352	507	66	3925
2000	9932	1085	159	11178

Tabulka 4: MPRGP - počet jednotlivých kroků algoritmu

4.2 Ořezání zvukového záznamu

Ořezání zvukového záznamu je důležité pro mnoho audio aplikací. Příliš vysoká amplituda signálu totiž způsobuje nejenom horší kvalitu přehrávaného záznamu, ale může i poškodit přehrávací zařízení. Protože samotné ořezání zvukového záznamu také snižuje kvalitu záznamu, byly vytvořeny nové ořezávací algoritmy založené na zachování kvality vnímání (viz [9]).

Popis ořezávacího algoritmu

Vstupní audiosignál $\mathbf{x}[n]$ je rozdělen do oken obsahujících N vzorků s přesahem do dalšího okna o velikosti P vzorků. Potom se pro každé okno $\mathbf{x}_k$ provedou následující 3 kroky:

1. Vypočítá se globální maskovací práh $\mathbf{t}_k \in \mathbb{R}^{\frac{N}{2}+1}$ pomocí části MPEG-1 Layer-1 psychoakustického modelu 1 (viz [10]). Globální maskovací práh udává množství deformační energie (v dB), která může být maskována signálem.

2. Určí se optimální výstupní okno $\mathbf{y}_k^* \in \mathbb{R}^N$ jako výsledek následujícího minimalizačního problému:

$$\begin{aligned} \mathbf{y}_k^* &= \arg\min_{\mathbf{y}_k \in \mathbb{R}^N} f(\mathbf{y}_k) \quad \text{takové, že} \quad \mathbf{l} \leq \mathbf{y}_k \leq \mathbf{u} \\ &= \arg\min_{\mathbf{y}_k \in \mathbb{R}^N} \frac{1}{2N} \sum_{i=0}^{N-1} \mathbf{w}_k(i) |\mathbf{Y}_k(e^{j\omega_i}) - \mathbf{X}_k(e^{j\omega_i})|^2 \\ &\qquad \text{takové, že} \quad \mathbf{l} \leq \mathbf{y}_k \leq \mathbf{u} \end{aligned} \tag{28}$$

3. Na optimální výstupní okno se aplikuje lichoběžníkové okno tak, aby se po sečtení výstupních oken okna v přesahové zóně křížila. Toto zajistí souvislost výsledného signálu.

Po provedení těchto 3 kroků se optimální výstupní okna $\mathbf{y}_k^*$ sečtou do výstupního audiosignálu $\mathbf{y}[n]$.

V minimalizačním problému (28) vektory $\mathbf{l}$ a $\mathbf{u}$ udávají minimální a maximální amplitudu výsledného signálu, $\omega_i = (2\pi i)/N$ značí diskrétní frekvenční proměnnou, $\mathbf{X}_k(e^{j\omega_i})$ a $\mathbf{Y}_k(e^{j\omega_i})$ jsou diskrétní frekvenční komponenty $\mathbf{x}_k$ a $\mathbf{y}_k$ a $\mathbf{w}_k(i)$ jsou váhy váhové funkce vnímání definované pomocí vztahu

$$\mathbf{w}_k(i) = \begin{cases} = 10^{-\alpha \mathbf{t}_k(i)} & \text{pro } 0 \leq i \leq \frac{N}{2} \\ = 10^{-\alpha \mathbf{t}_k(i)} & \text{pro } \frac{N}{2} < i \leq N-1 \end{cases} \tag{29}$$

kde kompresní parametr α náleží do rozmezí $0.04 - 0.06$.

Formulaci (28) můžeme přepsat na standartní úlohu kvadratického programování, který můžeme řešit algoritmy z předchozí kapitoly následovně:

$$\begin{aligned} \mathbf{y}_k^* &= \arg\min_{\mathbf{y}_k \in \mathbb{R}^N} \frac{1}{2} (\mathbf{y}_k - \mathbf{x}_k)^H \mathbf{D}^H \mathbf{W}_k \mathbf{D} (\mathbf{y}_k - \mathbf{x}_k)^H \quad \text{takové, že} \quad \mathbf{l} \leq \mathbf{y}_k \leq \mathbf{u} \\ &= \arg\min_{\mathbf{y}_k \in \mathbb{R}^N} \frac{1}{2} \mathbf{y}_k^H \mathbf{D}^H \mathbf{W}_k \mathbf{D} \mathbf{y}_k - (\mathbf{D}^H \mathbf{W}_k \mathbf{D} \mathbf{x}_k)^H \mathbf{y}_k \quad \text{takové, že} \quad \mathbf{l} \leq \mathbf{y}_k \leq \mathbf{u} \end{aligned}$$

kde $\mathbf{D} \in \mathbb{C}^{N \times N}$ je matice diskrétní Fourierovy transformace definována jako

$$\mathbf{D} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{-j\omega_1} & e^{-j\omega_2} & \dots & e^{-j\omega_{N-1}} \\ 1 & e^{-j\omega_2} & e^{-j\omega_4} & \dots & e^{-j\omega_{2(N-1)}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ +1 & e^{-j\omega_{N-1}} & e^{-j\omega_{2(N-1)}} & \dots & e^{-j\omega_{(N-1)(N-1)}} \end{pmatrix}$$

a $\mathbf{W}_k \in \mathbb{R}^{N \times N}$ je diagonální matice s pozitivními vahami $\mathbf{w}_k(i), i = 0, 1, \dots, N-1$ na diagonále, definovanými v (29).

Parametry a výsledky úlohy

V naší úloze se budeme zabývat jen druhým krokem ořezávacího algoritmu, tedy minimalizačním problémem. Použijeme audiosignál o velikostech vzorků

$$N = 100, 200, 500, 1000, 2000,$$

zvolíme parametr $\alpha = 0.04$ a požadovanou přesnost řešení $\epsilon = 10^{-2}$. Použijeme omezení

$$\begin{aligned} \mathbf{l} &= (-0.4, -0.4, \ldots, -0.4)^T, \\ \mathbf{u} &= (0.4, 0.4, \ldots, 0.4)^T. \end{aligned}$$

Dosažené výsledky jsou zobrazeny v tabulkách 5 a 6. V tabulce 5 jsou zapsány celkové počty násobení maticí $\mathbf{A}$ u jednotlivých algoritmů. Výsledky Barzilai-Borweinova algoritmu a MPRGP algoritmu jsou přibližně stejné. Zároveň jsou oba algoritmy pro tuto úlohu mnohem rychlejší než pro úlohu struny. Nejhorších výsledků znovu dosahuje metoda největšího spádu, která vůbec nekonverguje k řešení v povoleném počtu iterací.

N	Metoda největšího spádu	BB algoritmus	MPRGP algoritmus
100	∞	61	32
200	∞	45	37
500	∞	35	51
1000	∞	25	57
2000	∞	25	69

Tabulka 5: Počet násobení maticí $\mathbf{A}$

V tabulce 6 je uvedeno rozvržení provedených kroků MPRGP algoritmu a celkový počet iterací algoritmu. Oproti příkladu struny převládá v tomto případě Expanzní krok.

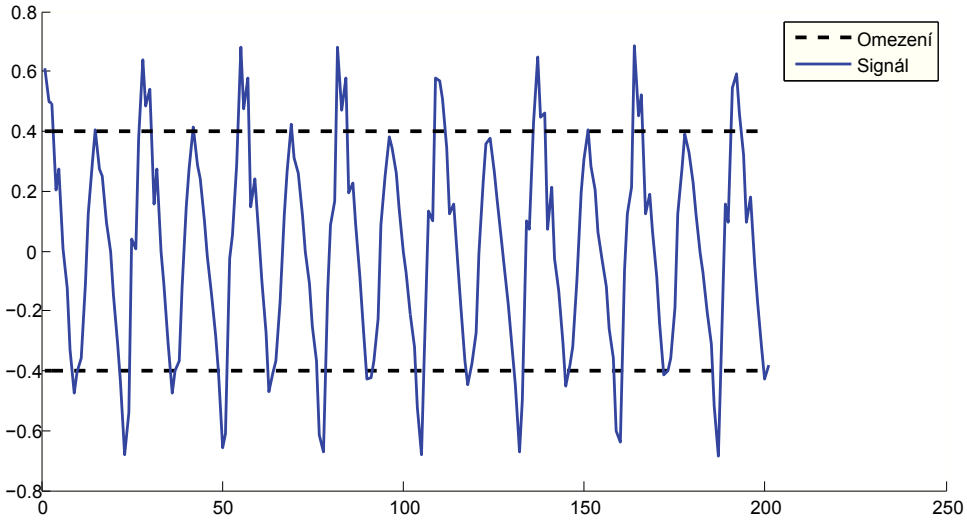
N	Sdružený gradientní krok	Expanzní krok	Proporcionální krok	Celkem
100	3	14	0	17
200	2	17	0	19
500	2	24	0	26
1000	2	27	0	29
2000	2	33	0	35

Tabulka 6: MPRGP - počet jednotlivých kroků algoritmu

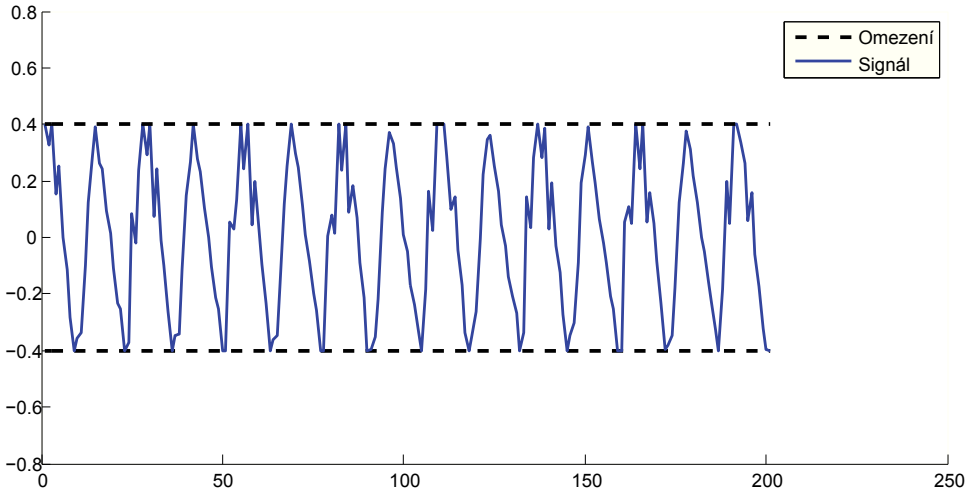
Na následujících obrázcích si ukážeme, jak vypadá samotné ořezání signálu. Použijeme audiosignál o velikosti $N = 200$ vzorků, znovu zvolíme $\alpha = 0.04$ a budeme požadovat přesnost řešení $\epsilon = 10^{-2}$. Použijeme stejné omezení, tedy

$$\begin{aligned} \mathbf{l} &= (-0.4, -0.4, \ldots, -0.4)^T, \\ \mathbf{u} &= (0.4, 0.4, \ldots, 0.4)^T. \end{aligned}$$

Na obrázku 16 vidíme původní, neořezaný signál vzhlem k zadaným omezením. Obrázek 17 pak zobrazuje signál po ořezání algoritmem MPRGP.



Obrázek 16: Původní signál



Obrázek 17: Ořezaný signál

5 Závěr

Cílem této práce bylo popsat vybrané algoritmy určené k řešení úlohy kvadratického programování a aplikovat je pro úlohy ořezání audiosignálu a modelování průhybu struny.

Ve druhé kapitole jsme si zavedli pojem minimalizační úloha bez omezení a popsali jsme metodu největšího spádu, metodu sdružených gradientů a Barzilai-Borweinův algoritmus používáné pro její řešení. Ve třetí kapitole jsme se zabývali úlohou minimalizace s lineárními nerovnostními omezeními, uvedli jsme, co to znamená ortogonální projekce na množinu a popsali jsme MPRGP algoritmus vyvinutý prof. Dostálem. Ve čtvrté kapitole jsme algoritmy metody největšího spádu, Barzilai-Borweina a MPRGP vyzkoušeli na úlohách modelování průhybu struny a ořezání zvukového signálu a vzájemně je porovnali.

Na tuto práci je možné navázat v mnoha inženýrských úlohách, které často vedou na úlohy kvadratického programování.

6 Literatura

[1] V. Vondrák, L. Pospíšil, *Numerické metody I*. Matematika pro inženýry 21. století (reg. č. CZ.1.07/2.2.00/07.0332), 2011.

[2] Z. Dostál, Lineární Algebra. Skriptum VŠB-TU Ostrava , 2000.

[3] J. Barzilai, J.M.Borwein, Two-point step size gradient methods. IMA Journal of Numerical Analysis 8, 1988.

[4] J. Bouchala, Matematika 3 pro bakalářské studium, 2000.

[5] Z. Dostál, Optimal quadratic programming algorithms, with applicattions to variational inequalities, 2009.

[6] Z. Dostál, L. Pospíšil, Optimal iterative QP and QPQC algorithms, 2013.

[7] Z. Dostál, J. Schöberl, Minimizing Quadratic Functions Subject to Bound Constraints with the Rate of Convergence and Finite Termination, 2005.

[8] T. Kozubek, T. Brzobohatý, M. Jarošová, V. Hapla, A. Makropoulos, Lineární Algebra s Matlabem. Matematika pro inženýry 21. století (reg. č. CZ.1.07/2.2.00/07.0332), 2012.

[9] B. Defraene, T. Waterschoot, H. J. Ferreau, M. Diehl, M. Moonen, Perception-Based Clipping of Audio Signals, 2010

[10] ISO/IEC, 11172-3 Information technology - Coding of moving pictures and associated audio for digital storage media at up about 1.5 Mbit/s - Part 3: Audio, 1993

A Přílohy

Příloha na CD/DVD:

babycry.wav
bba.m
mns.m
mprgp.m
PrikladBB.m
PrikladBB_2.m
PrikladMNS.m
PrikladMNS_2.m
PrikladMPRGP_2.m
PrikladMSG.m
struna1.m
struna1BBA.m
struna1MNS.m
struna1MPRGP.m
struna2.m
struna2BBA.m
struna2MNS.m
struna2MPRGP.m
zvuk.m
zvukBBA.m
zvukMNS.m
zvukMPRGP.m